Path-Following Method to Determine the Field of Values of a Matrix with High Accuracy

SIAM J. MATRIX ANAL. APPL.
Vol. 39, No. 4, pp. 1726–1749

PATH-FOLLOWING METHOD TO DETERMINE THE FIELD OF VALUES OF A MATRIX WITH HIGH ACCURACY*

SÉBASTIEN LOISEL† AND PETER MAXWELL‡

Abstract. We describe a novel and efficient algorithm for calculating the field of values boundary, $\partial\mathrm{W}(\cdot)$, of an arbitrary complex square matrix: the boundary is described by a system of ordinary differential equations which are solved using Runge–Kutta (Dormand–Prince) numerical integration to obtain control points with derivatives, then finally Hermite interpolation is applied to produce a dense output. The algorithm computes $\partial\mathrm{W}(\cdot)$ both efficiently and with low error. Formal error bounds are proven for specific classes of matrix. Furthermore, we summarize the existing state of the art and make comparisons with the new algorithm. Finally, numerical experiments are performed to quantify the cost-error trade-off between the new algorithm and existing algorithms.

Key words. field of values, numerical range, Johnson's algorithm, Runge–Kutta, Dormand–Prince, parametrized eigenvalue problem, eigenvalue perturbation, eigenvalue crossing

AMS subject classifications. 15A18, 15A60, 15B57, 65F15, 65F30

DOI. 10.1137/17M1148608

1. Introduction.

For a complex matrix $A \in \mathbb{C}^{n\times n}$, the *field of values* or *numerical range* $\mathrm{W}(A)$ is the image of the unit sphere under the Rayleigh quotient of the matrix:

$$\mathrm{W}(A) := \{\, \boldsymbol{x}^* A\boldsymbol{x} \,:\, \boldsymbol{x} \in \mathbb{C}^n,\ \boldsymbol{x}^*\boldsymbol{x} = 1 \,\}.$$

The field of values encloses the set of eigenvalues and can be used in a similar manner as the set of eigenvalues, e.g., to estimate the magnitude of matrix functions [10]; see also [44, p. 167]. Our interest is in finite-dimensional complex matrices. However, $\mathrm{W}(A)$ can also be defined in a similar manner when A is an operator acting on a general Hilbert space [21, p. 1].

The problem of calculating the field of values boundary, $\partial\mathrm{W}(A)$, has a long history, with the canonical algorithm surely being that of Johnson [26], but see also [36] and [28], [14]. We now briefly outline Johnson's algorithm. For a given matrix $A \in \mathbb{C}^{n\times n}$, define $H(e^{it}A) := \frac{1}{2}(e^{it}A + e^{-it}A^*)$. $H(e^{it}A)$, being Hermitian, has real eigenvalues. Let $\lambda_{\max}(t)$ denote the largest eigenvalue of $H(e^{it}A)$, and let $\boldsymbol{u}_{\max}(t)$ denote a corresponding unit eigenvector. Then, Johnson's function $\zeta(t) = \boldsymbol{u}_{\max}(t)^* A \boldsymbol{u}_{\max}(t)$ is a parametrization of $\partial\mathrm{W}(A)$ for $t \in [0, 2\pi)$. Although $\partial\mathrm{W}(A)$ is continuous with at most n points of discontinuous first derivative (see subsection 2.2), continuity does not necessarily hold for the parametrization $\zeta(t)$; indeed, this was alluded to in Johnson's paper [26], e.g., Theorem 3. If, for a given value of $t = t_k$, the eigenvalue $\lambda_{\max}(t_k)$ is nonsimple, then there are multiple linearly independent choices of $\boldsymbol{u}_{\max}(t_k)$, each, possibly, producing a different value of $\zeta(t_k)$. Therefore, there can be a jump discontinuity at t_k where $\zeta(t_k)$ is not well-defined but each of the one-sided limits $\zeta(t_k^+)$ and

*Received by the editors September 21, 2017; accepted for publication (in revised form) by M. A. Freitag September 13, 2018; published electronically November 27, 2018.
http://www.siam.org/journals/simax/39-4/M114860.html
Funding: The second author was funded by the Norwegian Research Council (FRINATEK), project 249740.
†Department of Mathematics, Heriot-Watt University, Edinburgh, Scotland EH14 4AS, UK (S.Loisel@hw.ac.uk).
‡Department of Energy and Process Engineering, Faculty of Engineering, NTNU–Norwegian University of Science and Technology, Trondheim NO-7491, Norway (peter.maxwell@ntnu.no).

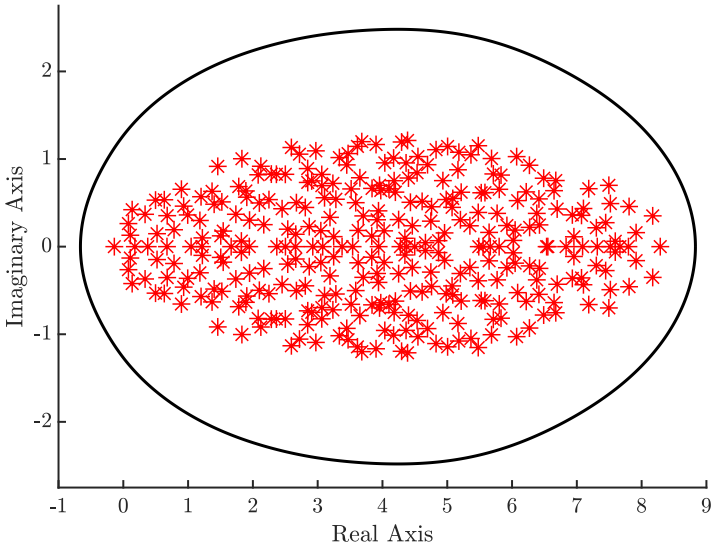


FIG. 1.1. *A plot of the field of values of the matrix* `A = delsq(numgrid('S',20))+ 0.1*randn(324)` *generated by the path-following algorithm. The red asterisks are the eigenvalues of A, and the solid black line is the boundary of* $\mathrm{W}(A)$. *(Figure in color online.)*

$\zeta(t_k^-)$ are well-defined; these t_k correspond to, possibly degenerate, flat segments on $\partial\mathrm{W}(A)$. We show in Theorem 7.2 that there can be only finitely many such t_k.

Johnson's algorithm is to sample $\zeta(t)$ at discrete values $t_k = (k-1)(2\pi/m)$, $k = 1, 2, \ldots, m$, and approximate $\partial\mathrm{W}(A)$ by the polygon whose vertices are $\{\zeta(t_k)\}_{k=1}^m$, where $m \geq 3$ is the number of vertices; we denote by $\hat{\zeta}(t)$ the piecewise linear approximation of $\zeta(t)$ produced by Johnson's algorithm. Provided that one has an accurate and robust eigenvalue solver, Johnson's algorithm is able to approximate $\partial\mathrm{W}(A)$ for sufficiently large m. The piecewise linear approximation $\hat{\zeta}(t)$ of $\zeta(t)$ is exact at the vertices, with $\mathcal{O}(m^{-2})$ error between the vertices. Therefore, achieving approximations of higher accuracy rapidly becomes prohibitively expensive. Several other algorithms and strategies have been proposed to calculate the field of values boundary $\partial\mathrm{W}(A)$, e.g., Marcus and Pesce [34], Braconnier and Higham [4], and Uhlig [46].

Our main new idea is to use a "path-following" algorithm to efficiently compute a high order near-interpolant $\hat{\xi}(t)$ of $\zeta(t)$. This is possible because, from eigenvalue perturbation analysis, one can show that $\zeta(t)$ is piecewise analytic. We derive a system of differential equations for the parametrized eigenvalue problem $H(e^{it}A)\boldsymbol{u}_{\max}(t) = \lambda_{\max}(t)\boldsymbol{u}_{\max}(t)$ and, using a suitable ODE solver with dense output, obtain a high order interpolant $\hat{\xi}(t)$ which approximates $\zeta(t)$. Care must be taken to stop and restart the ODE solver at points where $\zeta(t)$ is less smooth, which we do in an efficient manner. This feature makes our algorithm attractive compared to applying a high order interpolant directly to Johnson's algorithm: without knowledge of where $\zeta(t)$ is not smooth, the interpolant cannot achieve the anticipated accuracy. Johnson observes a similar result in his analysis at [26, p. 600]: he notes that "flat" portions of $\partial\mathrm{W}(A)$ were not considered and may lead to slow convergence. An example of the piecewise polynomial approximation of $\partial\mathrm{W}(A)$ produced by the path-following algorithm can be seen in Figure 1.1; this is a 324×324 matrix and the tolerance of

the ODE integrator was set to 10^{-10}.

The principal feature of our algorithm is that it computes $\mathrm{W}(A)$ more efficiently than previous algorithms when the sought tolerance is small. Our analysis and numerical experiments reveal that the path-following algorithm is more efficient than existing algorithms in practical settings at moderate accuracy and is also asymptotically faster. Indeed, at high accuracy, our algorithm is at least an order of magnitude faster. We also posit that our algorithm may have more generic applicability particularly in relation to Sturm–Liouville problems; this is discussed further in the conclusions.

1.1. Related results using computational perturbation methods. Our path-following algorithm inherently adopts a perturbative or *continuation* approach. For context and without any aim of completeness, we make a very terse summary of related results that the reader may find useful.

Lui [32] derives continuation methods based on inverse iteration and Lanczos for calculation of pseudospectra. A homotopy method for solving the eigenproblem of large sparse real nonsymmetric matrices is given by Lui, Keller, and Kwok in [33].

In [19], Guglielmi and Overton describe an algorithm to calculate the pseudospectral abscissa and pseudospectral radius by using the rightmost eigenvalue of a sequence of rank-1 updates $B_{k+1} = A+\varepsilon \boldsymbol{y}_k \boldsymbol{x}_k^*$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the leading "RP-compatible" left and right eigenvectors of B_k. In [17], [18], Guglielmi and Lubich observe that Guglielmi and Overton's algorithm is an iterative algorithm on the manifold of normalized rank-1 matrices, $\mathcal{M} = \{\, \boldsymbol{u}\boldsymbol{v}^* \,:\, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{C},\, \|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1 \,\}$. By reframing the problem as a continuous dynamical system on $\mathcal{M}$, a differential equation can be used to solve for $E(t)$ on $\mathcal{M}$ such that the real part of the leading eigenvalue of $A + \varepsilon E(t)$ tends to the pseudospectral abscissa. In [29], Kressner and Vandereycken extend the results of Guglielmi and Overton by using a subspace acceleration method.

In [3], Beyn and Thümmler create a predictor-corrector continuation method for invariant subspaces (invariant pairs) of the quadratic eigenvalue problem with large sparse matrices depending on a single real parameter; they include several relevant and instructive practical applications. In [2], Beyn, Effenberger, and Kressner analyze invariant pairs for nonlinear eigenproblems which are entrywise holomorphic functions in the eigenvalue parameter and describe a pseudoarclength predictor-corrector technique for continuation of the invariant pairs.

Due to the philosophical relevance to the work herein, we also mention Sirković and Kressner's result in [42] for approximating the smallest eigenvalue of a parameter-dependent Hermitian matrix.

1.2. Organization of the paper. Our paper is organized as follows. In section 2, we briefly recall the basic properties of the field of values and then review existing algorithms in section 3. In section 4, we give a brief overview of our new algorithm and clarify some properties of the Johnson parametrization in section 5. We fully specify the path-following algorithm in section 6. We provide analyses of expected eigenvalue crossings and error estimates in section 7. In section 8, we numerically compare the running time and accuracy of our new algorithm against several existing algorithms, including Johnson's algorithm. We end with some conclusions.

2. Basic properties of the field of values. The fundamental properties and results concerning the field of values are more than adequately described in other sources [21], [24], [28], [14]. Nevertheless, we make use of some of these results herein, which shall be restated briefly without proof. The properties and results summarized

in this section are in the context of our restricted finite-dimensional case only and may not hold in a more general setting.

DEFINITION 2.1 (Hermitian and skew-Hermitian parts of a matrix). *For notational convenience, we define*

$$H(A) := \frac{1}{2}(A + A^*) \quad \textit{and} \quad S(A) := \frac{1}{2}(A - A^*)$$

as the Hermitian and skew-Hermitian parts of the matrix, A. A matrix A can be written as $A = H(A) + S(A)$.

DEFINITION 2.2 (extremal eigenvalues). *For Hermitian* $K \in \mathbb{C}^{n\times n}$, *we denote the least and greatest eigenvalues by* $\lambda_{min}(K)$ *and* $\lambda_{max}(K)$. *For scalar t and square matrix A, we define* $\lambda_{max}(t)$ *as the greatest eigenvalue of* $H(e^{it}A)$.

2.1. Fundamental properties. Let $A, B \in \mathbb{C}^{n\times n}$.

Property 2.3.

(a) The eigenvalues of A are contained within $\mathrm{W}(A)$, $\sigma(A) \subseteq \mathrm{W}(A)$;
(b) the field of values is linear with respect to scaling and translation, $\mathrm{W}(\alpha A + \beta I) = \alpha\mathrm{W}(A) + \beta$ for all $\alpha, \beta \in \mathbb{C}, \alpha \neq 0$;
(c) unitary similarity invariance, $\mathrm{W}(U^*AU) = \mathrm{W}(A)$ for all unitary $U \in \mathbb{C}^{n\times n}$;
(d) $\mathrm{W}(A)$ is compact (image of a compact set under a continuous function) [21, p. 4], [24, p. 1];
(e) $\mathrm{W}(A)$ is a convex set, cf. the Toeplitz–Hausdorff theorem [23], [43], and also [11], [12], [20], [24], [37], [39];
(f) $\partial\mathrm{W}(A)$, where $n = 2$ is a, possibly degenerate, ellipse, cf. elliptical range theorem as described in [21, p. 3], Theorem 1.3.6a of [24, p. 23], and [31];
(g) $\mathrm{W}(A) \subset \mathbb{R}$ if and only if A is Hermitian, in other words $\mathrm{W}(A)$ is an interval on the real line whenever A is Hermitian and, furthermore, $\Re[\mathrm{W}(A)] = \mathrm{W}(H(A)) = [\lambda_{\min}(H(A)), \lambda_{\max}(H(A))]$;
(h) if A is normal, then $\mathrm{W}(A) = \mathrm{Co}(\sigma(A))$, in other words if A is normal, then the field of values is the convex hull of the eigenvalues (the converse is, in general, not true for $n > 4$, cf. [25], [35]);
(i) for $A_1 \oplus A_2 = \left[\begin{smallmatrix} A_1 & 0 \\ 0 & A_2 \end{smallmatrix}\right], A_1 \in \mathbb{C}^{n_1\times n_1}, A_2 \in \mathbb{C}^{n_2\times n_2}$, then $\mathrm{W}(A_1 \oplus A_2) = \mathrm{Co}(\,\mathrm{W}(A_1) \bigcup \mathrm{W}(A_2)\,)$, the field of values of a direct sum of matrices is the convex hull of the union of the field of values of those matrices; and
(j) for a multi-index, $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, define the principal submatrix of A as $A(\boldsymbol{\alpha})$ being the rows and columns of A indexed by $\boldsymbol{\alpha}$ then $\mathrm{W}(A(\boldsymbol{\alpha})) \subseteq \mathrm{W}(A)$.

2.2. "Corners" or "sharp points" of $\partial\mathrm{W}(A)$. Consider $A \in \mathbb{C}^{n\times n}$. $\partial\mathrm{W}(A)$ is smooth except possibly at a finite number of so-called "sharp points" or "corners" which have nonunique tangents [24, p. 50], [21, p. 19] or, equivalently, where $\partial\mathrm{W}(A)$ is not a differentiable arc [12]. A sharp point of $\partial\mathrm{W}(A)$ must be an eigenvalue of A although the converse does not necessarily hold; cf. Theorem 1 of [12] and Theorem 1.5-5 of [21, p. 20]. Therefore, there are at most n sharp points on $\partial\mathrm{W}(A)$. Herein, we adopt the formal definition of a sharp point used in [24, p. 50].

DEFINITION 2.4 (sharp point). *A point* $\alpha \in \partial\mathrm{W}(A)$ *is a sharp point if there are* t_1 *and* t_2 *with* $0 \leq t_1 < t_2 < 2\pi$ *for which*

$$\Re(e^{it}\alpha) = \max\{\Re(\beta) : \beta \in \mathrm{W}(e^{it}A)\} \quad \textit{for all } t \in (t_1, t_2).$$

Sharp points can be characterized by Theorem 1.6.6 in [24] or Theorem 5 in [8].

THEOREM 2.5. *Let $A \in \mathbb{C}^{n\times n}$ and $\alpha \in \partial \mathrm{W}(A)$. Then α is a sharp point if and only if A is unitarily similar to $\alpha I_m \oplus A_1$ with $A_1 \in \mathbb{C}^{(n-m)\times(n-m)}$ and $\alpha \notin \mathrm{W}(A_1)$. In this case α is the intersection of two flat line portions on $\partial \mathrm{W}(A)$.*

3. Existing algorithms. There are only a handful of existing algorithms for computing the field of values boundary of a general complex matrix. The $n = 2$ case can be trivially handled using the elliptical range theorem, Property 2.3 (f).

3.1. Johnson's algorithm. There is almost a de facto standard, published by Johnson in 1978 [26], for computing the field of values of a general complex square matrix. The concept behind this method—taking advantage of the convexity property Property 2.3 (e) of $\mathrm{W}(A)$ by successively applying a scalar rotation Property 2.3 (b) to the matrix A then using the Hermitian projection property Property 2.3 (g) to bound the field of values set between the least and greatest eigenvalues—has been expressed at least as far back as Murnaghan's terse result in 1932 [36] and Kippenhahn's more comprehensive results in 1951 [28], [14]. However, Johnson's result is the first instance of the method being used to create a convergent computational algorithm.

We will make use of this methodology, so we shall briefly recap the result from [26]. Note that the Hermitian projection property Property 2.3 (g) states that $\Re[\mathrm{W}(A)] = [\lambda_{\min}(H(A)), \lambda_{\max}(H(A))]$. Denote by $\boldsymbol{u}_{\max}$ an associated unit eigenvector for $\lambda_{\max}$. The line $L = \{\, \lambda_{\max}(H(A)) + is \,:\, s \in \mathbb{R} \,\}$ defines a vertical support line tangent to $\partial \mathrm{W}(A)$. The intersection $L \cap \partial \mathrm{W}(A)$ may not be a single point as $\mathrm{W}(A)$ can have a straight line or "flat" segment along its boundary. Furthermore, where L and $\partial \mathrm{W}(A)$ intersect is also not necessarily on the real axis. The point

$$p := \boldsymbol{u}_{\max}^* A \boldsymbol{u}_{\max} \in L \cap \partial \mathrm{W}(A) \tag{3.1}$$

is on $\partial \mathrm{W}(A)$ with Figure 3.1a depicting this arrangement. Note, furthermore, that $e^{-it}\mathrm{W}(e^{it}A) = \mathrm{W}(A)$. By successively applying a rotation by angle t_k to A as $e^{it_k}A$ for $k = \{1, \ldots, K\}$, collecting the p_k points for corresponding t_k, and calculating the convex hull will produce a piecewise linear inner boundary approximation for $\partial \mathrm{W}(A)$. Johnson takes this further by calculating an outer boundary using the supporting hyperplanes. This constrains $\partial \mathrm{W}(A)$ between an inner and outer boundary so that an error bound can be calculated. This arrangement is depicted in Figure 3.1b.

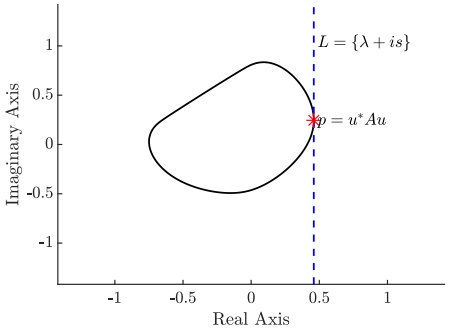
3.2. Marcus–Pesce algorithm. An alternative method was described by Marcus and Pesce in 1987 [34] which involves using multiple random orthonormal matrix compressions of the form $V^*AV \in \mathbb{C}^{2\times 2}$ for $V \in \mathbb{C}^{n\times 2}$. For a matrix $V = \begin{bmatrix}\boldsymbol{v}_1 \, \boldsymbol{v}_2 \, \ldots \, \boldsymbol{v}_k\end{bmatrix} \in \mathbb{C}^{n\times k}$, where the $\boldsymbol{v}_i \in \mathbb{C}^n$ vectors form an orthonormal basis for V and $A \in \mathbb{C}^{n\times n}$, the *(matrix) compression* of A is defined as

$$A_V := V^*AV,$$

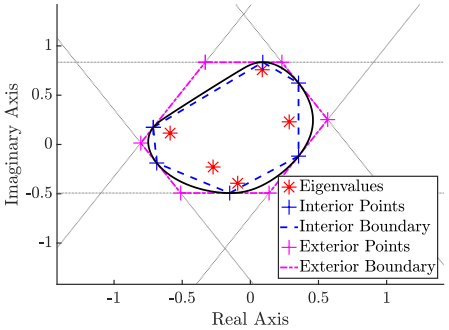
where $A_V \in \mathbb{C}^{k\times k}$. By using Property 2.3 (c) and Property 2.3 (j), it can be seen that $\mathrm{W}(V^*AV) \subseteq \mathrm{W}(A)$. Using the elliptical range theorem, Property 2.3 (f), when $k = 2$, then $\mathrm{W}(V^*AV)$ is a, possibly degenerate, ellipse. In Theorem 1 of [34], it is proven that the field of values is equal to the union over all 2×2 matrix compressions using $V = [\boldsymbol{v_1}\, \boldsymbol{v_2}]$, where $\boldsymbol{v_1}, \boldsymbol{v_2}$ are real and orthonormal,

$$\mathrm{W}(A) = \bigcup_V \mathrm{W}(V^*AV).$$

Marcus and Pesce's computation strategy is essentially to pick a reasonable number of random real orthonormal vector pairs, compute the resulting set of elliptical

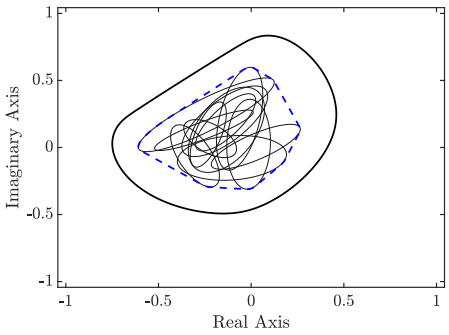


(a) Bounding of $\partial W(A)$ using Hermitian part of matrix. Shown without rotation ($t = 0$). The vertical bounding support line is in dashed blue; the p point is marked with a red asterisk; and $\partial W(A)$ in solid black.

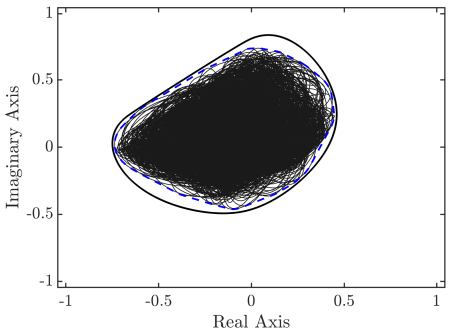


(b) Illustration of an inner and outer Johnson boundary for six rotation increments. Inner boundary is in dashed blue; outer boundary is in dash-dotted magenta; outer supporting hyperplanes in dotted faint grey; eigenvalues of A are shown with red asterisks; and exact $\partial W(A)$ is in solid black.

FIG. 3.1. *Plots showing how the Johnson algorithm bounds* $\partial W(A)$. *(Figure in color online.)*



(a) 12 ellipses: the Marcus–Pesce boundary is a poor approximation of $\partial W(A)$.

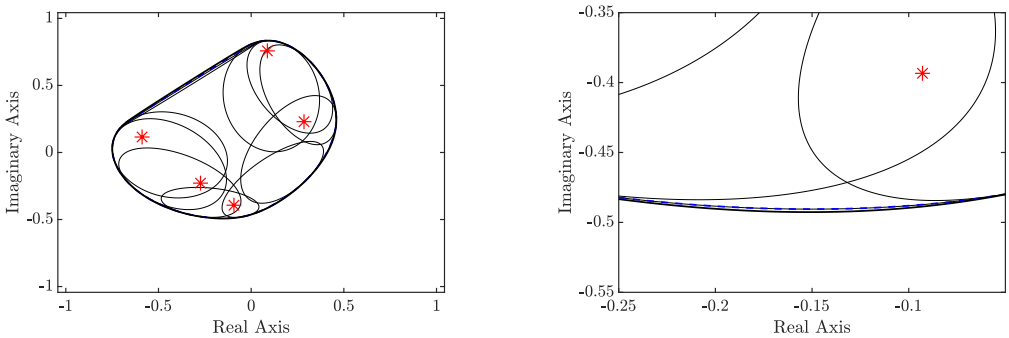


(b) 1000 ellipses: the Marcus–Pesce boundary is still showing high error.

FIG. 3.2. *Figures akin to Uhlig's figure* [46, *Fig.* 5] *showing Marcus and Pesce's randomly generated eigenvectors for matrix compressions. The matrix compression ellipses are shown with thin black lines; the generated boundary in dashed blue; and exact* $\partial W(A)$ *in solid black. (Figure in color online.)*

field of values sets, and then plot (or compute the convex hull and plot). This works reasonably well for very small matrices but selecting vectors at random fails to approximate the boundary accurately for matrices with $n > 4$; see Figure 3.2.

3.3. Uhlig's optimization of Marcus–Pesce. The strategy adopted by Uhlig [46] is faster and more accurate compared to the Marcus–Pesce method. Instead of picking random pairs of real orthonormal vectors, Uhlig proposes first calculating a small number of ordered boundary points $p_k = \boldsymbol{u}_k^* A \boldsymbol{u}_k$ using Johnson's method and then using successive eigenvector pairs from this list for the matrix compressions so that each ellipse is more likely to constrain the boundary. In particular, Uhlig starts by constructing a so-called *Bendixson box*: for a particular t_k, the greatest and least eigenvalues from both the Hermitian and skew-Hermitian parts are used, which creates



(a) Uhlig's method quickly constrains the boundary: the generated boundary in dashed blue is close to $\partial \mathrm{W}(A)$.

(b) A zoomed-in plot so that the generated and true boundaries can be distinguished.

FIG. 3.3. *Calculation of* $\partial \mathrm{W}(A)$ *using Uhlig's method for a random matrix* A *of size* $n = 5$. *Matrix compression ellipses can be seen internal to* $\mathrm{W}(A)$. $\partial \mathrm{W}(A)$ *is shown in solid black whereas the boundary generated from the matrix compressions is in dashed blue. Eigenvalues are denoted with red asterisks. (Figure in color online.)*

a rectangular bounding box from the four initial eigenpairs. This initial list of four vectors is then expanded by the judicious choice of further Johnson calculations, each returning two additional vectors—corresponding to the desired $\lambda_{\min}$ and $\lambda_{\max}$—to merge into the list. Uhlig proposed four rationales to choose these additional points [46, sect. 2.1]. Assuming the order of the vectors is maintained properly to represent either a clockwise or anticlockwise ordering, then once a suitable number of vectors are collated, matrix compressions can be performed along the boundary. The algorithm successively creates matrices V_k by picking the kth and $(k+1)$th vector pair, with k running over the full list, orthonormalizes the pair, calculates the matrix compression A_{V_k}, and then calculates a suitable number of points on the resulting field of values ellipse. Finally, a convex hull is calculated on all the collected ellipse points to achieve a boundary approximation as shown in Figure 3.3.

3.4. Braconnier–Higham approach. Braconnier and Higham applied a specific implementation of the Lanczos algorithm for computing the field of values using Johnson's method [4]. Their work focuses on optimizing the eigenproblem solves and applying a *continuation* method. In this context, continuation means that when performing an eigenproblem solve using a Krylov subspace method one can often use the eigenvector from the previous result as the starting vector for the next computation in order to reduce the number of iterations required.

4. Algorithm overview. The strategy behind the new algorithm is to determine $\partial \mathrm{W}(A)$ by "tracking" the dominant eigenpair of the Hermitian part of $e^{it}A$ as a function of t in an analogous manner to Johnson's method as described in subsection 3.1. This can be expressed as

$$H(e^{it}A)\boldsymbol{u}(t) = \lambda(t)\boldsymbol{u}(t), \tag{4.1}$$

where $\{\boldsymbol{u}(t), \lambda(t)\}$ is the dominant eigenpair for t within a suitable interval. The next step is to take the derivative of (4.1) and rearrange

$$H(e^{it}A)\boldsymbol{u'}(t) - \lambda'(t)\boldsymbol{u}(t) - \lambda(t)\boldsymbol{u'}(t) = -iS(e^{it}A)\boldsymbol{u}(t). \tag{4.2}$$

By inspection, $H(e^{it}A)$ is elementwise comprised of analytic functions of the single real variable t and is thus differentiable in the usual matrix sense. Due to an elegant result of Rellich [40, Chap. 1] for a Hermitian matrix A whose elements are analytic functions of some real parameter t, say, the eigenvalues can be considered analytic functions of t if suitably ordered. Furthermore, there exists an orthonormal basis of eigenvectors which are also elementwise comprised of analytic functions of t, and therefore the eigenvectors are differentiable in the usual vector sense.

Whilst the previous results are general with respect to eigenvalue multiplicity, a more prosaic problem arises: when attempting to track the dominant eigenvalue through a nonsimple eigenvalue—a so-called *level-crossing* [1, p. 350] or *exceptional point* [27, p. 74]—the ordering may impose that the tracked eigenvalue is no longer the dominant eigenvalue after the crossing. Therefore, it is assumed that the eigenvalue $\lambda(t)$ is simple for $t \in (t_1, t_2)$. In other words, the algorithm cannot process intervals within which the dominant eigenvalue is nonsimple for some values of t.

Note that (4.2) is underdetermined. We choose the eigenvectors of $H(e^{it}A)$ to be orthonormal and use the identity

$$\boldsymbol{u}(t)^*\boldsymbol{u}(t) = 1 \tag{4.3}$$

as the required additional constraint. Differentiating (4.3), we obtain $\Re[\boldsymbol{u}(t)^*\boldsymbol{u}'(t)] = 0$. Note that $\boldsymbol{u}(t)$ is a unit eigenvector and so is $e^{i\theta}\boldsymbol{u}(t)$ for any "phase parameter" $\theta \in \mathbb{R}$. In order to fix the phase parameter so that $\boldsymbol{u}(t)$ is uniquely defined, we further impose the ODE $\Im[\boldsymbol{u}(t)^*\boldsymbol{u}'(t)] = 0$. Combining these two ODEs together produces $\boldsymbol{u}(t)^*\boldsymbol{u}'(t) = 0$ (requiring that the derivative, $\boldsymbol{u}'(t)$, be orthogonal to $\boldsymbol{u}(t)$). Including the constraint $\boldsymbol{u}(t)^*\boldsymbol{u}'(t) = 0$ gives

$$\begin{cases} H(e^{it}A)\boldsymbol{u}'(t) - \lambda'(t)\boldsymbol{u}(t) - \lambda(t)\boldsymbol{u}'(t) = -iS(e^{it}A)\boldsymbol{u}(t), \\ \boldsymbol{u}(t)^*\boldsymbol{u}'(t) = 0, \end{cases} \tag{4.4}$$

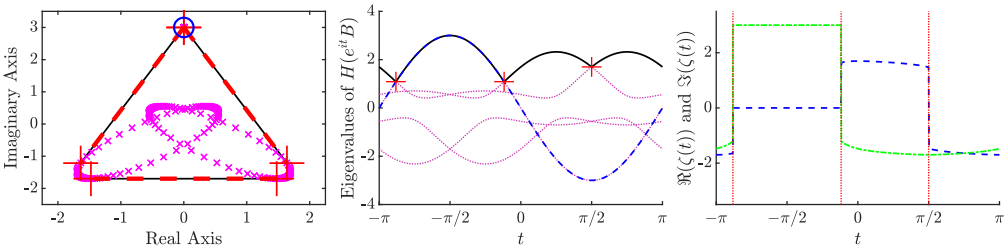
which after rearranging can be written in matrix form as

$$\begin{bmatrix} H(e^{it}A) - \lambda(t)I & -\boldsymbol{u}(t) \\ -\boldsymbol{u}(t)^* & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}'(t) \\ \lambda'(t) \end{bmatrix} = \begin{bmatrix} -iS(e^{it}A)\boldsymbol{u}(t) \\ 0 \end{bmatrix}. \tag{4.5}$$

As intimated by an anonymous referee, our proposed method is an ODE on a manifold. A comparison may be made with [17].

The general approach, within a given interval (t_1, t_2) with midpoint t_{mid}, is to calculate the dominant $\{\boldsymbol{u}(t_{\mathrm{mid}}), \lambda(t_{\mathrm{mid}})\}$ pair by solving an eigenvalue problem, and then use (4.5) to numerically integrate $\{\boldsymbol{u}'(t), \lambda'(t)\}$ along t both forwards and backwards to generate a solution $\{\boldsymbol{u}(t), \lambda(t)\}$ for the whole interval. The numerical integrator must be able to detect whether the dominant eigenvalue becomes nonsimple. Using the parlance of numerical integration, this is denoted as an *event*; cf. [22, Chap. II.3]. The integrator calculates along intervals until it either reaches the end point or encounters an event, whereupon it stops. The integration can be restarted on the other side of the nonsimple eigenvalue and the two solutions then joined together. Since the output from the Dormand–Prince integrator [13, p. 23] has both values and derivatives, Hermite interpolation can be performed to create a dense output.

5. Observations on the Johnson parametrization $\zeta(t)$. In the introduction, we described Johnson's function $\zeta(t) = \boldsymbol{u}_{\max}(t)^* A \boldsymbol{u}_{\max}(t)$ as a parametrization of $\partial\mathrm{W}(A)$. From a computational standpoint, this is the most natural choice, yet it is not as regular as might be expected.



(a) $\partial\mathrm{W}(B)$ boundary shown with a black line. The Johnson points corresponding to all eigenvalues—not only the dominant eigenvalue as is usual—are shown with small magenta crosses. The sharp point at $3i$ is marked with a blue circle. The flat segments where $\lambda_{\max}(e^{it_k}B)$ is nonsimple are shown in thick dashed red with plus signs indicating the extent of those segments.

(b) Eigenvalues of $H(e^{it}B)$ shown in thin dotted magenta. The dominant $\lambda_{\max}(e^{it}B)$ is shown with a black line. Nonsimple eigenvalues of $\lambda_{\max}(e^{it}B)$ are denoted with red plus signs; at these points the first derivative of $\lambda_{\max}(e^{it}B)$ is discontinuous due to a change of order of eigenvalue. The sharp point—a normal eigenvalue—is shown with a dashed-blue overlay.

(c) The real part of $\zeta(t)$ shown in dashed blue; the imaginary part of $\zeta(t)$ shown in dash-dotted green. The positions of the nonsimple eigenvalues of $\lambda_{\max}(e^{it}B)$ are indicated with vertical red dotted lines. Note the constant region where $\Re(\zeta(t)) = 0$ and $\Im(\zeta(t)) = 3$, which corresponds to the sharp point in (a) at $3i$.

FIG. 5.1. *Plots of $\partial\mathrm{W}(B)$ and the eigenvalues of $H(e^{it}H)$ for example matrix B. Note that on* (b)*, the normal eigenvalue path can be written αe^{it}, and when it is the dominant eigenvalue, it corresponds directly with Definition* 2.4*: it maps the whole interval to the single sharp point on $\partial\mathrm{W}(B)$. Observe that the nonsimple eigenvalue at $\pi/2$ generates the flat segment on the bottom of the $\partial\mathrm{W}(B)$ and that the two other nonsimple eigenvalues generate the flat segments at the sides of $\partial\mathrm{W}(B)$. (Figure in color online.)*

Due to the possibility of nonsimple eigenvalues on $\lambda_{\max}(H(e^{it}A))$, $\zeta(t)$ can be undefined at finitely many t_k which correspond to flat segments on $\partial\mathrm{W}(A)$. We prove this in Lemmas 5.1 and 5.2 and Theorem 5.3 below; also see Theorem 1 in [7]. Furthermore, $\lambda_{\max}(H(e^{it}A))$ may have discontinuous first derivative whenever it is a nonsimple eigenvalue due to the change in order of dominant eigenvalues.

Using Definition 2.4, it can be immediately observed that $\zeta(t)$ is not, in general, injective: for a sharp point $\alpha \in \partial\mathrm{W}(A)$, $\zeta(t)$ maps some interval $t \in (t_1, t_2)$, $t_1 < t_2$ to the single point α. Again from Definition 2.4 and perhaps counterintuitively, $\lambda_{\max}(H(e^{it}A))$ is continuous for $t \in (t_1, t_2)$. Furthermore, from Theorem 2.5, it is clear that the existence of one or more sharp points necessarily requires the existence of nonsimple eigenvalues on $\lambda_{\max}(H(e^{it}A))$.

These properties are illustrated in Figure 5.1 for a carefully constructed matrix, B, composed of the direct product of two 2×2 ellipse matrices and one normal point.

LEMMA 5.1. *Let $A \in \mathbb{C}^{n\times n}$ be such that for some $t_0 \in [0, 2\pi)$, the greatest eigenvalue $\lambda_{max}(H(e^{it_0}A))$ has algebraic and geometric multiplicity $m > 1$, and, furthermore, that $\lambda_{max}(H(e^{it}A))$ is simple for t in a neighborhood of t_0. Define the rotated matrix as $A_0 := e^{it_0}A$. Choose $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$ to be m distinct linearly independent unit eigenvectors of $H(A_0)$ for $\lambda_{max}(H(A_0))$ and denote the associated Johnson points of A_0 as $\{\, p_j = \boldsymbol{u}_j^* A_0 \boldsymbol{u}_j \,\}_{j=1}^m$. Furthermore, let $\mu_j = \Im(p_j)$ and assume the ordering of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$ is such that $\mu_1 \le \cdots \le \mu_m$.*

(A) *Then $\zeta(t_0)$ is well-defined if and only if $\mu_1 = \mu_m$.*

(B) *If $\mu_1 \ne \mu_m$, then t_0 in the Johnson parametrization corresponds to a flat segment on $\partial\mathrm{W}(A_0)$ which is a vertical line in the complex plane.*

Proof. (A) Note that by definition, $\Re(p_i) = \Re(p_j)$ for $i, j \in 1 \dots m$. Then

$$\zeta(t_0) \text{ is well-defined } \iff p_1 = \cdots = p_m \iff \mu_1 = \mu_m.$$

(B) Assume $\mu_1 \neq \mu_m$. Then the line $L = \{\, \Re(p_1) + i(s\mu_1 + (1-s)\mu_m) \,:\, s \in [0,1] \,\}$ is a vertical flat segment on $\partial \mathrm{W}(A_0)$. From inspection, all other p_j also lie on L. ∎

LEMMA 5.2. *Let $A \in \mathbb{C}^{n\times n}$ be such that there is a flat segment on $\partial \mathrm{W}(A)$. Let $t_0 \in [0, 2\pi)$ be the angle so the flat segment on $\partial \mathrm{W}(e^{it_0}A)$ is arranged to be a vertical line rightmost to $\mathrm{W}(e^{it_0}A)$. Then $\lambda_{max}(H(A_0))$ has multiplicity $m > 1$.*

Proof. Let p_1 and p_2 be two distinct points on the flat segment of $\partial \mathrm{W}(A_0)$. Then, by using Lemma 1.5.7 from [24], we can choose distinct unit eigenvectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ of $H(A_0)$ such that $p_1 = \boldsymbol{u}_1^* A_0 \boldsymbol{u}_1$ and $p_2 = \boldsymbol{u}_2^* A_0 \boldsymbol{u}_2$ and $\lambda_{\max}(H(A_0)) = \Re(p_1) = \Re(p_2)$. Therefore, the eigenspace for $\lambda_{\max}(H(A_0))$ has dimension at least 2, so multiplicity $m > 1$. ∎

THEOREM 5.3. *Let $A \in \mathbb{C}^{n\times n}$. A flat segment exists on $\partial \mathrm{W}(A)$ if and only if there exists a $t_0 \in [0, 2\pi)$ such that $\lambda_{max}(H(e^{it_0}A))$ has multiplicity $m > 1$ and for which there exists distinct unit eigenvectors, $\boldsymbol{u}_1, \boldsymbol{u}_2$ with $\Im(\boldsymbol{u}_1^* e^{it_0} A \boldsymbol{u}_1) \neq \Im(\boldsymbol{u}_2^* e^{it_0} A \boldsymbol{u}_2)$.*

Proof. The result follows from Lemmas 5.1 and 5.2. ∎

6. Full algorithm specification. Recapping from section 4, the essential feature of the algorithm is to calculate a high order near-interpolant $\hat{\xi}(t)$ of $\zeta(t)$ by numerically integrating a system of ODEs to obtain the path of the dominant eigenvalue and eigenvector of $H(e^{it}A)$, from which an approximation of $\zeta(t)$ can be calculated.

DEFINITION 6.1 (matrix M_A). *We define the matrix-valued function*

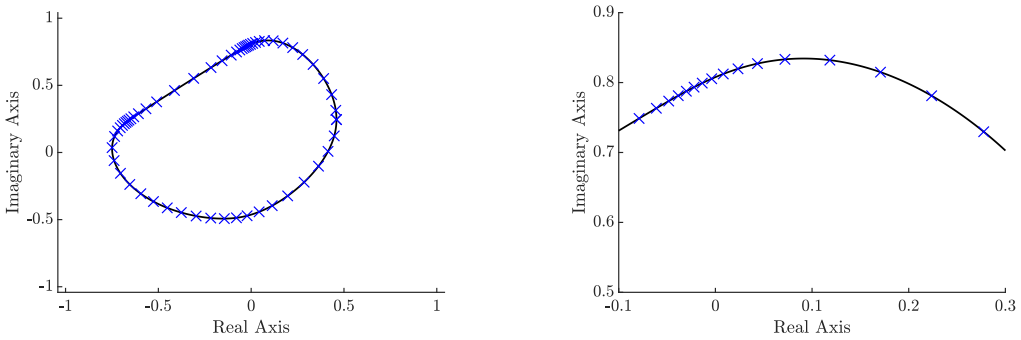
$$M_A(z, \boldsymbol{u}, \lambda) := \begin{bmatrix} H(zA) - \lambda I & -\boldsymbol{u} \\ -\boldsymbol{u}^* & 0 \end{bmatrix}.$$

For notational convenience, we define $\boldsymbol{v}(t) := [\ \boldsymbol{u}(t)^*\ \lambda(t)^*\ ]^*$ and $\boldsymbol{f}(t, \boldsymbol{v}(t)) := \boldsymbol{v}'(t) = [\ \boldsymbol{u}'(t)^*\ \lambda'(t)^*\ ]^*$, where unless otherwise stated, $\lambda(t)$ is understood to be the dominant eigenvalue parametrized by t, and $\boldsymbol{u}(t)$ is an eigenvector for $\lambda(t)$. Given a candidate $\boldsymbol{v}(t)$, a linear solver can be used to obtain $\boldsymbol{v}'(t)$ from (4.5),

$$\boldsymbol{f}\left(t, \begin{bmatrix} \boldsymbol{u}(t) \\ \lambda(t) \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{u}'(t) \\ \lambda'(t) \end{bmatrix} = \overbrace{\begin{bmatrix} H(e^{it}A) - \lambda(t) I & -\boldsymbol{u}(t) \\ -\boldsymbol{u}(t)^* & 0 \end{bmatrix}^{-1}}^{M_A^{-1}(e^{it}, \boldsymbol{u}(t), \lambda(t))} \begin{bmatrix} -iS(e^{it}A)\boldsymbol{u}(t) \\ 0 \end{bmatrix}. \tag{6.1}$$

In subsection 6.1 we show that for simple $\lambda_{\max}(t)$, the matrix M_A is invertible and so $\boldsymbol{f}$ is well-defined.

We use the Dormand–Prince RK5(4)7M method [13, p. 23] with the Hermite interpolation strategy of Shampine [41, p. 148] (an extra row in the Butcher tableau is used to calculate a midpoint without any additional $\boldsymbol{f}(\cdot)$ evaluations). This integrator produces a solution which is 5th order accurate, i.e., whose error term is $\mathcal{O}(h^5)$ with a degree 4 Hermite interpolant for its dense output. For a given integration step on an interval $[t_k, t_{k+1}]$ with midpoint t_{mid}, we obtain output vectors $\{\widetilde{\boldsymbol{v}}_k, \widetilde{\boldsymbol{v}}'_k, \widetilde{\boldsymbol{v}}_{k+1/2}, \widetilde{\boldsymbol{v}}_{k+1}, \widetilde{\boldsymbol{v}}'_{k+1}\}$, where tilde quantities represent the solutions obtained from the numerical integration. For brevity, we write $\widetilde{\boldsymbol{v}}_k$ to mean $\widetilde{\boldsymbol{v}}(t_k)$, and $\widetilde{\boldsymbol{v}}_{k+1/2}$ is taken to mean $\widetilde{\boldsymbol{v}}(t_{\mathrm{mid}})$.



(a) Plot at full scale. (b) Zoomed-in plot.

FIG. 6.1. *Plot of $\partial \mathrm{W}(A)$ approximation created by new algorithm for random $n = 5$ matrix at tolerance 10^{-7}. Black continuous boundary is the Hermite interpolant $\hat{\xi}(t)$. The blue crosses show the locations of the control points, ξ_k, from each stage of the Dormand–Prince integration. (Figure in color online.)*

Each output vector is $(n+1)$-dimensional but we are only interested in the calculation of a 1-dimensional parametrized approximation $\hat{\xi}(t)$ of $\zeta(t)$. We use Johnson's expression, (3.1), to calculate the (inner) boundary points ξ_k, $\xi_{k+1/2}$, and ξ_{k+1},

$$\xi_k = \widetilde{\boldsymbol{v}}_k^* \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \widetilde{\boldsymbol{v}}_k = \widetilde{\boldsymbol{u}}_k^* A \widetilde{\boldsymbol{u}}_k. \tag{6.2}$$

In a similar fashion, the derivatives ξ_k' and ξ_{k+1}' can be calculated as

$$\xi_k' = \boldsymbol{f}(t_k, \widetilde{\boldsymbol{v}}_k)^* \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \widetilde{\boldsymbol{v}}_k + \widetilde{\boldsymbol{v}}_k^* \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{f}(t_k, \widetilde{\boldsymbol{v}}_k). \tag{6.3}$$

Thus, we obtain a set of ξ_k, ξ_k', and $\xi_{k+1/2}$ upon which Hermite interpolation can be performed. The obtained solution curve $\hat{\xi}(t)$ is a scalar-valued piecewise polynomial function of t. After calculation of all the ξ_k, ξ_k', and $\xi_{k+1/2}$ points to a given tolerance, any number of evaluations of $\hat{\xi}(t)$ can be computed efficiently by performing 1-dimensional interpolation. This results in an appreciable $\mathcal{O}(n)$ gain in performance when the dimension n is large. This arrangement is demonstrated in Figure 6.1.

Events are detected by using the method in subsection 6.2. The algorithm is described by the pseudocode in Algorithm 6.1.

6.1. Linear solver. For $\boldsymbol{f}$ to be well-defined, $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ must be an invertible matrix.

DEFINITION 6.2 (matrix D_A). *Let $\eta_1(t) \le \eta_2(t) \le \cdots \le \eta_n(t)$ be the eigenvalues of $H(e^{it}A)$, $t \in [0, 2\pi)$. For a given eigenpair $\{\boldsymbol{u}(t), \lambda(t)\}$ of $H(e^{it}A)$, let j be such that $\eta_j(t) = \lambda(t)$. Let $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}\}$ be the multi-index*

$$\alpha_k = \begin{cases} k \text{ for } k < j, \\ k+1 \text{ for } k \ge j. \end{cases} \tag{6.4}$$

Algorithm 6.1. Field of values path-following algorithm.

Initialize $L = \{J\}$, where J is some interval of length 2π
while L is not empty: **do**
  Remove an interval $[t_{\min}, t_{\max}]$, with midpoint t_{mid}, from L.
  Compute the exact dominant eigenpair $\{\boldsymbol{u}_{\max}(t), \lambda_{\max}(t)\}$ of (4.1) for $t = t_{\text{mid}}$.
  Using initial values $t = t_{\text{mid}}$ and $[\ \boldsymbol{u}_{\max}(t)^*\ \lambda_{\max}(t)^*\ ]^*$:
    numerically integrate $\boldsymbol{f}(\cdot)$ forward on $[t_{\text{mid}}, t_{\max}]$ by solving (6.1), and
    numerically integrate $\boldsymbol{f}(\cdot)$ backward on $[t_{\text{mid}}, t_{\min}]$ by solving (6.1).
  if $q_A(\cdot)$ event triggered (Definition 6.4) **then**
    Calculate event location(s) using $r_A(\cdot)$ (Definition 6.5).
    Denote by $[t_0, t_1]$ the interval of integration until the event(s).
    Insert $[t_{\min}, t_{\max}] \setminus [t_0, t_1]$ into L.
  end if
  Compute the relevant $\xi_k, \xi_k', \xi_{k+1/2}$ using (6.2) and (6.3).
end while
The curve $\hat{\xi}(t)$ is computed using Hermite interpolation of the corresponding $\xi_k, \xi_k', \xi_{k+1/2}$.

We denote by $D_A(t, \boldsymbol{u}(t), \lambda(t))$ the unitary transform of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ such that

$$D_A(t, \boldsymbol{u}(t), \lambda(t)) := U(t)^*\, M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))\, U(t)$$
$$= \begin{bmatrix} \eta_{\alpha_1}(t) - \lambda(t) & & & & \\ & \ddots & & & \\ & & \eta_{\alpha_{n-1}}(t) - \lambda(t) & & \\ & & & 0 & -1 \\ & & & -1 & 0 \end{bmatrix}.$$

LEMMA 6.3. *Assume that $\boldsymbol{u}(t)$ and $\lambda(t)$ are close to the solution curve. Denote by $\eta_1(t) \le \eta_2(t) \le \cdots \le \eta_n(t) = \lambda(t)$ the eigenvalues of $H(e^{it}A)$, and $D_A(t, \boldsymbol{u}(t), \lambda(t))$ the unitary transform of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ as defined in Definition 6.2. Furthermore, assume $\eta_{n-1}(t) < \eta_n(t)$. Then, $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ is invertible.*

Proof. By observation from $D_A(t, \boldsymbol{u}(t), \lambda(t))$, the eigenvalue of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ with the smallest magnitude is either $\eta_{n-1}(t) - \eta_n(t)$ if that quantity is small or ± 1 from the lower-right 2×2 block. ∎

The case $\eta_{n-1}(t) = \eta_n(t)$ is an event that stops integration, as described in the next subsection.

6.2. Event detection within Dormand–Prince integrator. We require a method to detect whether there has been an event within an integration step and, if so, to calculate its location. From Lemma 6.3, we see that the eigenvalues of $H(e^{it}A)$ and $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ are closely related, by a shift of $\lambda(t)$, except possibly for the two eigenvalues ± 1 of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$.

DEFINITION 6.4 (event function $q_A(\cdot)$). *Given an eigenpair $\{\boldsymbol{u}(t), \lambda(t)\}$ of $H(e^{it}A)$, define the function $q_A(t, \boldsymbol{u}(t), \lambda(t))$ to be equal to the smallest magnitude eigenvalue of $D_A(t, \boldsymbol{u}(t), \lambda(t))$ excluding the ± 1 eigenvalues arising from the lower-right 2×2 block of D_A. If $q_A(t, \boldsymbol{u}(t), \lambda(t)) < 0$, then $\lambda(t)$ is the dominant eigenvalue; if $q_A(t, \boldsymbol{u}(t), \lambda(t)) =$*

0, *then* $\lambda(t)$ *is nonsimple; and, if* $q_A(t, \boldsymbol{u}(t), \lambda(t)) > 0$, *then* $\lambda(t)$ *is no longer dominant.*

At each integration step, if $q_A(t, \boldsymbol{u}(t), \lambda(t)) < 0$, then no event has occurred. Otherwise, an event has occurred and we must determine the location, which is equivalent to finding a zero of $q_A(\cdot)$. We assume the integration step size small enough so that only one event may occur within a step.

DEFINITION 6.5 (event location $r_A(\cdot)$). *Given continuous eigenpair* $\{\boldsymbol{u}(t), \lambda(t)\}$, *for* $t \in [t_1, t_2]$ *and* $q_A(t_1, \boldsymbol{u}(t_1), \lambda(t_1))\ q_A(t_2, \boldsymbol{u}(t_2), \lambda(t_2)) < 0$, *we define* $r_A(t_1, t_2, \boldsymbol{u}(t), \lambda(t))$ *as equal to* $t_0 \in (t_1, t_2)$ *such that* $q_A(t_0, \boldsymbol{u}(t_0), \lambda(t_0)) = 0$.

6.2.1. Calculating $q_A(\cdot)$ using the inverse iteration with Aitken acceleration. $q_A(t, \boldsymbol{u}(t), \lambda(t))$ can be computed from $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ by performing an inverse iteration and orthonormalizing with respect to $[\,\boldsymbol{u}(t)^*\ 0\,]^*$ and $[\,0\ \ldots\ 0\ 1\,]^*$ at each iteration. This guarantees that we will compute the smallest magnitude eigenvalue of $\eta_{\alpha_1}(t) - \lambda(t), \ldots, \eta_{\alpha_{n-1}}(t) - \lambda(t)$. The inverse iteration is defined by $\boldsymbol{v}^{(k+1)} = M_A^{-1}\boldsymbol{v}^{(k)}$, where $\boldsymbol{v}^{(0)}$ is drawn randomly according to a standard normal distribution. We also perform Aitken delta-squared acceleration [38, p. 275], [6, p. 399] so that convergence is quadratic. We make the following definitions:

- $\rho_k = \frac{[\boldsymbol{v}^{(k)}]^* \boldsymbol{v}^{(k)}}{[\boldsymbol{v}^{(k)}]^* \boldsymbol{v}^{(k+1)}}$ (the Rayleigh quotient);
- $\mu_k = \rho_{k+2} - \frac{(\rho_{k+2} - \rho_{k+1})^2}{\rho_{k+2} - 2\rho_{k+1} + \rho_k}$ (Aitken acceleration); and
- $e_k = |\mu_k - \rho_k|$ (the approximate error).

We stop the inverse iteration if any of the following conditions are satisfied, where c_0 is a suitable constant (in our implementation $c_0 = 1.1$):

- $\mu_k + c_0 e_k < 0$, in which case, we conclude that λ is indeed the dominant eigenvalue of $H(e^{it}A)$;
- $\mu_k - c_0 e_k > 0$, here we conclude that λ is no longer the dominant eigenvalue of $H(e^{it}A)$ and an event has occurred; or
- a division by zero has occurred in the calculation of μ_k or e_k is smaller than some tolerance, in which case the two largest eigenvalues of $H(e^{it}A)$ are almost exactly equal.

To determine whether an event has occurred within an integration step, we do not need an accurate approximation of $q_A(\cdot)$; just the sign is sufficient. So, only a few iterations of the inverse iteration suffice in this situation. In the case where an event has occurred, we must find the location. Assuming that the step size δt of the ODE solver is small, then too shall $\eta_{n-1}(t) - \lambda(t)$ be small and hence only a few iterations are required.

One might expect that a subspace method such as Lanczos would be more efficient than the inverse iteration for this purpose. Indeed, MATLAB's `eigs()` implements Lanczos and was tested as a replacement for the inverse iteration with a variety of tolerance and subspace dimension choices. It was consistently slower than our custom implementation of the inverse iteration. This is likely due to the small number of iterations required.

6.2.2. Calculating the event location $r_A(\cdot)$. The precise location of an event can be found by using a nonlinear root finding algorithm applied to the $q_A(\cdot)$ event function. We use MATLAB's `fzero()`. An anonymous referee points out that `fzero()` can have a moderately large computational cost. This calculation is necessary because we must find, to high accuracy, the location of any nonsimple eigenvalues; without this information, the path-following algorithm may become unstable. The al-

gorithm underlying `fzero()` is Brent's modification of Dekker's algorithm; see [5] and [15]. It is guaranteed to converge and for continuously differentiable functions (assuming negligible rounding error), the convergence is superlinear. It posed no problems in our numerical experiments.

7. Analysis. The performance characteristics of the algorithm are essentially predicated on two properties: the number of eigenvalue problems that must be solved and the error bounds for a Runge–Kutta step of size h. These are characterized in the subsections below.

7.1. Expectation of eigenvalue crossings. In sections 4 and 6, it was explained that numerical integration can only proceed along an arc whilst the dominant eigenvalue $\lambda_{\max}(t)$ is simple. If a nonsimple eigenvalue is encountered, a new arc must be computed on the other side of the eigenvalue crossing thus requiring another eigenvalue solve. Since eigenvalue solves are the most computationally expensive step in the path-following algorithm, it is desirable to know a priori the expected number of eigenvalue crossings and also an upper bound.

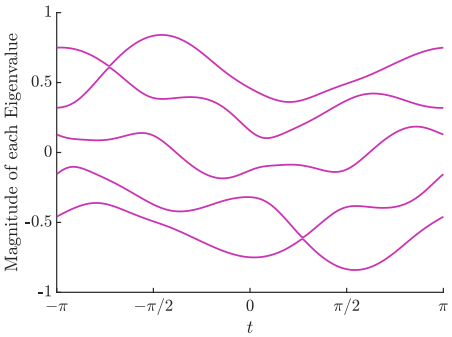
Recalling our earlier description of the problem in (4.1), we are concerned with the eigenvalues of a Hermitian matrix parameterized by a single real variable, $H(e^{it}A) = (e^{it}A + e^{-it}A^*)/2$. For a general random Hermitian matrix parametrized by a single real variable, von Neumann and Wigner argued in their celebrated 1929 paper [48], [47] that, without special structure, eigenvalue crossings are highly unlikely to occur. Further explanation can be found in [30, p. 140]. We provide numerical results in subsection 8.3 which support von Neumann and Wigner's heuristic result that for randomly generated matrices, eigenvalue crossings do not occur. However, the result only holds for randomly generated matrices. Normal matrices, for example, exhibit eigenvalue crossings, and so it is desirable to have a strict upper bound. We prove in Theorem 7.2 that the number of eigenvalue crossings is upper bounded as $2n(n-1)$. This situation is illustrated in Figures 7.1 and 7.2. Matrix J is defined as

$$J = J_A \oplus J_B = \begin{bmatrix} -1+i & 1 & 0 & 0 \\ 0 & -1+i & 0 & 0 \\ 0 & 0 & 1-i & 1 \\ 0 & 0 & 0 & 1-i \end{bmatrix}. \tag{7.1}$$

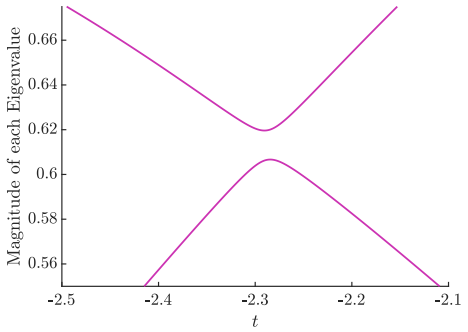
DEFINITION 7.1 (discriminant and resultant of $H(zA)$). *Let $A \in \mathbb{C}^{n\times n}$. Consider the family of matrices $H(zA) = (zA + z^{-1}A^*)/2$, parametrized by $z \in \mathbb{C} \setminus \{0\}$. The characteristic polynomial can be written $p_{H(zA)}(\lambda) = p_n(z)\lambda^n + \cdots + p_0(z)$. The discriminant of $H(zA)$ can be written in terms of the resultant* [9], [16],

$$\mathrm{Disc}_\lambda\left(p_{H(zA)}\right) := (-1)^{n(n-1)/2}\frac{1}{p_n(z)}\,\mathrm{Res}\left(p_{H(zA)}(\lambda), \frac{\partial}{\partial\lambda}p_{H(zA)}(\lambda)\right).$$

Let $q(z)$ be the Laurent polynomial obtained by writing the resultant as the deter-

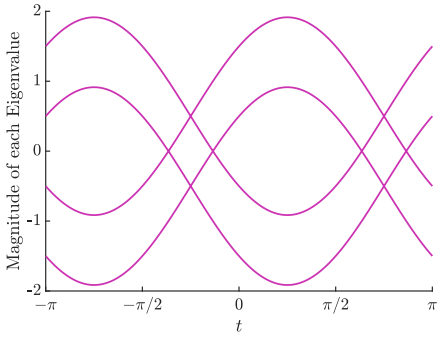


(a) Eigenvalues of $H(e^{it}A)$ for a random example matrix A, parametrized by t. First appearances suggest the eigenvalues cross.

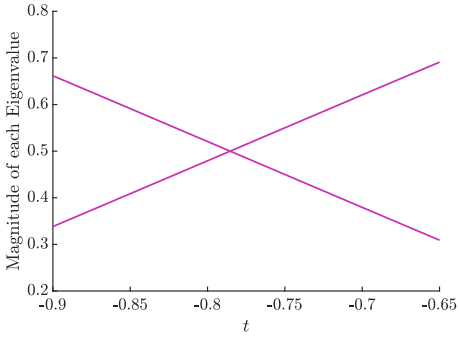


(b) Zooming in shows that the eigenvalues actually do not cross—an avoided crossing.

FIG. 7.1. *Eigenvalue avoided crossings.*



(a) Eigenvalues of $H(e^{it}J)$ for matrix J parametrized by t.



(b) This time the eigenvalues do indeed cross.

FIG. 7.2. *Eigenvalue level crossings.*

minant of a $(2n-1)\times(2n-1)$ *Sylvester matrix,*

$$q(z) = \det \underbrace{\begin{bmatrix} p_n(z) & \dots & \dots & p_0(z) & & \\ & \ddots & & & \ddots & \\ & & p_n(z) & \dots & \dots & p_0(z) \\ np_n(z) & \dots & p_1(z) & & & \\ & \ddots & & \ddots & & \\ & & \ddots & & \ddots & \\ & & & np_n(z) & \dots & p_1(z) \end{bmatrix}}_{Q}. \tag{7.2}$$

Note that the discriminant, $\mathrm{Disc}_\lambda\left(p_{H(zA)}\right)$, is zero if and only if the characteristic polynomial has a repeated root.

THEOREM 7.2. *Assume the family of matrices,* $H(zA)$*, discriminant, and resultant as defined in Definition* 7.1*. Assume that for some* $z \in \mathbb{C}\backslash 0$*, all the eigenvalues of* $H(zA)$ *are simple. Then, the eigenvalues of* $H(zA)$ *have algebraic multiplicity higher*

than 1 *for at most finitely many* $z \in \mathbb{C}$. *Moreover, if the number of such* z *is denoted as* l, *then* $l \leq 2n(n-1)$.

Proof. Since we are only interested in whether the discriminant is zero, the scalar factor can be dispensed with and only the resultant $q(z)$ need be considered.

Each $p_k(z)$ is a rational function of z, so the discriminant is a rational function of z. Furthermore, $q(z) = 0$ if and only if $p_{H(zA)}(\lambda)$ has repeated roots. Since $H(zA)$ has only simple eigenvalues for a certain value of z, $q(z)$ cannot be zero everywhere. By the fundamental theorem of algebra, $q(z)$ has at most finitely many roots.

Moreover, it can be seen that each $p_k(z)$ is a Laurent polynomial, $p_k(z) = a_m z^m + a_{m-1}z^{m-1} + \cdots + a_{-(m-1)}z^{m-1} + a_{-m}z^{-m}$ with $m = n - k$. By computing the determinant of Q, noting the maximum power of z in each term, it can be seen that the terms of $q(z)$ have maximum degree $n(n-1)$ in z. By applying the fundamental theorem of algebra again, $q(z)$ has at most $2n(n-1)$ roots and $l \leq 2n(n-1)$. ∎

7.2. Error estimates. For clarity, we denote by $\zeta(t)$ Johnson's parametrization of $\partial \mathrm{W}(A)$ and by $\hat{\xi}(t)$ the numerical approximation obtained from our Runge–Kutta solver. For certain classes of matrix, formal estimates are derived in relation to the step size, $h \ll 1$.

Consider the set $E \subset \mathbb{C}^{n\times n} \times \mathbb{C}$ of matrices and scalars $(A, z) \in \mathbb{C}^{n\times n} \times \mathbb{C}$ such that $H(zA)$ has repeated eigenvalues. This is precisely the set such that $q(z) = 0$, where $q(z)$ is the discriminant polynomial (7.2). Since $q(z)$ is a Laurent polynomial, we must have that E is Zariski closed and, in particular, E has Lebesgue measure zero. By the Fubini–Tonelli theorem, the restriction of E along almost any curve of real codimension 1 also has Lebesgue measure zero, so the set of $(A, z) \in \mathbb{C}^{n\times n} \times \mathbb{T}$ such that $H(zA)$ has repeated eigenvalues is of measure zero. In other words, repeated eigenvalues and events in the ODE solver are "exceptional"; for randomly generated matrices, such exceptional events will almost never occur.

DEFINITION 7.3 (arc). *Denote by* $m(z) \geq 1$ *the algebraic multiplicity of* $\lambda_{max}(H(zA))$. *We distinguish the following cases.*

1. *We say that* A *is* type 1 *if* $m(z) = 1$ *for all* $z \in \mathbb{T}$. *In this case, we define the only arc to be* $T_1 = \mathbb{T}$.
2. *We say that* A *is* type 2 *if* $m(z) > 1$ *for all* $z \in \mathbb{T}$. *This implies that the resultant* $q(z) = 0$ *for all* z, *so the set of all type* 2 *matrices is Zariski closed. We do not define any arcs in this situation.*
3. *We say that* A *is* type 3 *if it is neither type* 1 *nor type* 2. *We define* $\{z_k = e^{it_k}\}_{k=1}^K$ *as the finitely many values of* z *where* $m(z_k) > 1$. *The arcs are then of the form* $T_k = (t_k, t_{k+1})$ *for* $k = 1, \ldots, K$, *where* k *is understood modulo* K, *and* t *is understood modulo* 2π.

As noted in the introduction, $\zeta(t)$ *can be discontinuous at finitely many points* $\{t_k\}_{k=0}^K$, *corresponding to values of* t_k *where the eigenvalue* $\lambda_{max}(t_k)$ *is nonsimple. However, when* A *is either type* 1 *or type* 3, *the choice of* $\boldsymbol{u}_{\max}(t)$ *is unique up to rescaling inside each arc* $T_k = (t_k, t_{k+1})$. *The choice of* $\boldsymbol{u}_{\max}(t)$ *is nonunique at the vertices* $\{t_k\}$ *and at those points,* $\zeta(t_k)$ *is, in general, not well-defined.*

Consider an explicit Runge–Kutta method of order p for computing an approximation $\hat{\boldsymbol{v}}(t)$ of the dominant eigenpair $\boldsymbol{v}(t)$ for t in an arc. We analyze two cases: A being of *type* 1, and A being a normal matrix of *type* 3.

THEOREM 7.4. *If* A *is of type* 1, *then* $\|\hat{\boldsymbol{v}}(t) - \boldsymbol{v}(t)\| = \mathcal{O}(h^p)$.

Proof. Recall (6.1). From [40, Chap. 1], it can been seen that the eigenvalue

$\lambda(t)$ and the eigenvector $\boldsymbol{u}(t)$ are both analytic functions of t. Denote by $\eta_1(t) \leq \eta_2(t) \leq \cdots \leq \eta_n(t) = \lambda(t)$ the eigenvalues of $H(e^{it}A)$ and by $D_A(t, \boldsymbol{u}(t), \lambda(t))$ the unitary transform of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ as defined in Definition 6.2. Since, by assumption, $\lambda(H(e^{it}A))$ is simple for all t, then $\eta_{n-1}(t) < \eta_n(t)$, and so the eigenvalue of $M_A(e^{it}, \boldsymbol{u}(t), \lambda(t))$ with the smallest magnitude is either $\eta_{n-1}(t) - \eta_n(t)$ if that quantity is small or ± 1 from the lower-right 2×2 block of $D_A(t, \boldsymbol{u}(t), \lambda(t))$. Either way, since $\eta_{n-1}(t) - \eta_n(t)$ is nonzero for every $e^{it} \in \mathbb{T}$ and since $\mathbb{T}$ is compact, it must be that the modulus of the spectrum $|\sigma(M_A(e^{it}, \boldsymbol{v}_{\max}(t), \lambda_{\max}(t)))|$ is uniformly bounded below on $\mathbb{T}$ and hence $M_A^{-1}(e^{it}, \boldsymbol{u}(t), \lambda(t))$ is smooth in a neighborhood of the solution curve. From the standard theory of Runge–Kutta methods, e.g., Theorem II.3.4 in [22], we obtain the result. ∎

We now show that when A is normal, and $\partial \mathrm{W}(A)$ is, therefore, a polygon, our algorithm computes $\partial \mathrm{W}(A)$ exactly with no error. We begin with a technical lemma which shows that Runge–Kutta integrators preserve invariant subspaces.

LEMMA 7.5. *Consider a Runge–Kutta solver with S stages, with Butcher tableau B, c and step size h; in other words,*

$$\boldsymbol{y}^{(s)} = F\left(\boldsymbol{x}^{(0)} + h\sum_{j=1}^{s-1} B_{sj}\boldsymbol{y}^{(j)}\right) \text{ for } s = 1, \ldots, S \text{ and } \boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + h\sum_{k=1}^{S} c_k \boldsymbol{y}^{(k)}.$$

Say there is a matrix V such that $\boldsymbol{x} \in \operatorname{span} V \implies F(\boldsymbol{x}) \in \operatorname{span} V$, where $\operatorname{span} V$ denotes the column space of V. Then, if the initial data $\boldsymbol{x}^{(0)}$ is in the column span of V, all subsequent iterates $\boldsymbol{x}^{(k)}$ shall also be in the column space of V.

Proof. Clearly, $\boldsymbol{y}^{(0)} = F(\boldsymbol{x}^{(0)}) \in \operatorname{span} V$, and hence, by induction, $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(S)} \in \operatorname{span} V$. Finally, $\boldsymbol{x}^{(1)}$ is a linear combination of vectors in $\operatorname{span} V$. ∎

THEOREM 7.6. *Assume that A is a normal matrix with distinct eigenvalues and denote the solution generated by Algorithm 6.1 as $\hat{\xi}(t)$, the Runge–Kutta approximation of the Johnson function $\zeta(t)$. Then, $\hat{\xi}(t) = \zeta(t)$ for all $t \in \bigcup_k T_k$, where T_k are the arcs as defined in Definition 7.3, i.e., there is no error.*

Proof. Due to the spectral theorem, Property 2.3 (c), Property 2.3 (h), and Theorem 3 from [26], without loss of generality we may assume that $A = \operatorname{diag}(a_1, \ldots, a_n)$ with $a_1, \ldots, a_n \in \mathbb{C}$. Then, $H(e^{it}A) = \operatorname{diag}(\Re[e^{it}a_1], \ldots, \Re[e^{it}a_n])$, and so the differential equation is

$$\begin{bmatrix} \boldsymbol{u}'(t) \\ \lambda'(t) \end{bmatrix} = P(t)Q(t) \begin{bmatrix} \boldsymbol{u}(t) \\ \lambda(t) \end{bmatrix} = F\left(\begin{bmatrix} \boldsymbol{u}(t) \\ \lambda(t) \end{bmatrix}\right),$$

where

$$P(t) := \begin{bmatrix} \Re[e^{it}a_1] - \lambda(t) & & & & -u_1(t) \\ & \ddots & & & \vdots \\ & & \Re[e^{it}a_{n-1}] - \lambda(t) & & -u_{n-1}(t) \\ & & & \Re[e^{it}a_n] - \lambda(t) & -u_n(t) \\ -\overline{u}_1(t) & \cdots & -\overline{u}_{n-1}(t) & -\overline{u}_n(t) & 0 \end{bmatrix}^{-1}$$

and

$$Q(t) := \begin{bmatrix} i\Im[e^{it}a_1] & & & \\ & \ddots & & \\ & & i\Im[e^{it}a_n] & \\ & & & 0 \end{bmatrix}.$$

Clearly,

$$F\begin{pmatrix}0\\ \vdots\\ 0\\ *\\ *\end{pmatrix}=\begin{pmatrix}0\\ \vdots\\ 0\\ *\\ *\end{pmatrix},\quad \text{hence } V=\begin{bmatrix}0&0\\ \vdots&\vdots\\ 0&0\\ 1&0\\ 0&1\end{bmatrix}$$

is an invariant subspace for F. As per Algorithm 6.1, the initial value is the dominant eigenpair of $H(e^{it}A)\boldsymbol{u}(t)=\lambda(t)\boldsymbol{u}(t)$ so that $\lambda(t)=\Re[e^{it}a_n]$ and hence $\boldsymbol{u}(t)=[\,0\ \dots\ 0\ 1\,]^*=\boldsymbol{e}^{(n)}$. In particular, the initial point $\boldsymbol{x}^{(0)}$ is in the column space of V and hence, by Lemma 7.5, all the Runge–Kutta steps are in the column space of V. This means that $\hat{\boldsymbol{u}}_{\max}(t)$ is some multiple of $\boldsymbol{u}_{\max}(t)=\boldsymbol{e}^{(n)}$, and $\hat{\xi}(t)=\zeta(t)$. ∎

7.3. Discussion. We have proven the accuracy of our algorithm in the following cases:

- Type 1 matrices such that $\lambda_{\max}(H(zA))$ has multiplicity 1 for all $z\in\mathbb{T}$: our algorithm produces an approximation $\hat{\xi}(t)$ of $\zeta(t)$ that has $\mathcal{O}(h^p)$ accuracy.
- Type 2 matrices such that $\lambda_{\max}(H(zA))$ has multiplicity 2 or higher for all $z\in\mathbb{C}$: our algorithm does not work for those matrices.
- Type 3 matrices: if A is normal, then $\hat{\xi}(t)=\zeta(t)$ and the numerical solution is exact. We were not able to analyze the nonnormal case when nonsimple eigenvalues are present.

Another way of expressing the above is to state the amount of work required to obtain a solution to the accuracy of a specific tolerance, σ. An important component of this required work is the number of times we must perform an eigenproblem solve. For type 1 matrices, we need $\mathcal{O}(\sigma^{1/p})$ Runge–Kutta steps and the solution of one eigenproblem to produce initial conditions, while for normal matrices of type 3, we need $\mathcal{O}(1)$ steps of the Runge–Kutta solver and one eigenproblem solve per arc. Each Runge–Kutta step requires S_I linear solves to compute the S_I stages, plus a small number of linear solves S_L for event detection. Thus, for fixed $S=S_I+S_L$ and in the cases we have analyzed, we predict $\mathcal{O}(n_A(\sigma^{1/p}ST_L+T_E))$ work, where n_A is the number of arcs, T_L is the running time of a linear solve, and T_E is the running time of an eigenproblem solve. It is assumed that $T_E>T_L$. In our numerical experiments, n_A was often quite small. By comparison, Johnson's algorithm, which solves an eigenvalue problem at m values of t, requires $\mathcal{O}(mT_E)$ work. Since Johnson's algorithm has accuracy $\mathcal{O}(m^{-2})$, we arrive at the following running times:

- Our algorithm: $\mathcal{O}(n_A(\sigma^{-1/p}ST_L+T_E))$.
- Johnson's algorithm: $\mathcal{O}(\sigma^{-1/2}T_E)$.

As we can see, for small σ and when n_A is not too large, our algorithm is many orders of magnitude faster than Johnson's algorithm. Apart from the difference in order $\sigma^{-1/p}$ versus $\sigma^{-1/2}$, there is another trade-off that becomes obvious. Our algorithm requires many fewer eigenvalue solves, and in exchange we require some linear solves. This is especially appealing when A is a sparse matrix in very high dimensions, where sparse linear solvers may run faster than eigenvalue solvers.

Our algorithm does not apply to type 2 matrices, but this did not have a negative impact for our numerical experiments. We were also not able to completely analyze nonnormal matrices of type 3 (when $\lambda_{\max}(t)$ can be nonsimple). These limitations to our analysis are to be expected: even for the problem of computing eigenvalues, state-of-the-art eigenvalue solvers fail for some matrices.

8. Numerical tests. To provide a practical characterization of the new algorithm performance, numerical tests were performed to measure the error-cost trade-off in comparison with the existing algorithms.

8.1. Methodology for numerical tests. The most straightforward method to draw a fair comparison for practical use is to execute each algorithm within a range of parameters, record the runtime and error, and then compare the results. This ensures that what is measured is the trade-off between computational cost and error tolerance for each algorithm or parameter set. We considered a randomly generated complex matrix A of size $n = 250$ with $\|A\| = 1$ for the numerical tests.

In [26], Johnson estimates the error by calculating the area of the inner and outer boundaries which act as a lower and upper bound on the actual area; the difference can provide a suitable error measure for the number of boundary points calculated. However, if testing to high-precision, it is desirable to adopt an alternative error measure. Instead of using an area estimate, we attempt to calculate a "maximum distance" error between the exact boundary and the interpolant produced by each algorithm. In other words, we calculate the maximum over the minimum distance between all points on the exact field of values boundary and the candidate interpolant.

Let $\hat{\gamma}(\cdot)$ denote any generic parametrized boundary approximation (the interpolated output from a provided candidate algorithm). For a given exact Johnson point $p_k = \zeta(t_k)$ on $\partial \mathrm{W}(A)$, we define the error for a candidate algorithm at point p_k as

$$\epsilon(p_k) := \min_t \|p_k - \hat{\gamma}(t)\|,$$

and the error for K Johnson points, $p_k,\ k \in \{1, \ldots, K\}$ as

$$E(K) := \max_{k \in \{1,\ldots,K\}} \epsilon(p_k),$$

and we seek to estimate

$$E := \lim_{K\to\infty} E(K) = \lim_{K\to\infty} \max_{k \in \{1,\ldots,K\}} \min_t \|p_k - \hat{\gamma}(t)\|.$$

So that a valid comparison can be made with the analysis in section 7, we do not include in our measurements the time taken to compute the interpolant for each algorithm. For Johnson's algorithm and our path-following algorithm, these are essentially $\mathcal{O}(1)$ operations and are not interesting. For Uhlig's algorithm, matters are slightly more involved.

Using new notation for clarity, in Uhlig's algorithm, for each of the m_e compression ellipses one must calculate l_e boundary points, say. At the end of the algorithm's processing, a convex hull of the total $m_e l_e$ points must be calculated (the results of which can be piecewise linearly interpolated similar to Johnson's algorithm). Calculating points on the ellipse is very lightweight but calculating the convex hull is $\mathcal{O}(m_e l_e \log m_e l_e)$. Uhlig's algorithm as it stands also does not directly make use of the higher-order interpolation available from the ellipses.

With a view toward making our analysis clearer, we define an *Uhlig-lite* variant. Rather than calculating a set of boundary points from each ellipse, we instead only perform the eigensolves and matrix compressions. This is obviously always faster than Uhlig's full algorithm and so represents a lower bound on computational cost, i.e., Uhlig's full algorithm is always worse. To calculate $E(K)$, we calculate the minimum distance from each candidate p_k point to the nearest exact ellipse boundary (using a root solver). The effect of this is to determine the best-case asymptotic performance

of the matrix compression approach if perfect interpolation on the relevant ellipse could be done at no computational cost.

The eigenproblem solves for Johnson's algorithm, Uhlig's algorithm, and Uhlig-lite, and the path-following algorithm use QR decomposition. The Braconnier–Higham method is approximated by a version of Johnson's algorithm with MATLAB's `eigs()` implementation of Lanczos using the previous eigenvector for the new start vector (continuation). Many of the enhancements that Braconnier and Higham proposed for use in Lanczos (e.g., selective reorthonormalization and restarts) have been incorporated or improved upon by the ARPACK library that underpins MATLAB's `eigs()`, so we have implemented a modern version of Braconnier–Higham by combining Johnson's algorithm with MATLAB's `eigs()` solver.

The candidate algorithms are as follows: the path-following algorithm, Johnson's algorithm, Johnson's algorithm using Lanczos with continuation (as per Braconnier–Higham), Uhlig's algorithm, and the Uhlig-lite variant. For each algorithm, we approximate E to near machine precision by taking a suitably large K, picking each p_k randomly distributed over the boundary, and calculating the associated $\epsilon(p_k)$ values. This $E(K)$ will almost certainly not achieve the maximum E unless we are very lucky in our choice of p_k. So we therefore pick the $J \ll K$ largest $\epsilon(p_k)$ points labeling them $p_j,\ j \in \{1, \ldots, J\}$, choose a small arc around each p_j, create new p_k points along each arc and add to the list so that K grows, and finally re-evaluate $E(K)$ using these new points. This divide-and-conquer approach is repeated until successive $E(K)$ estimates converge to within machine precision, i.e., it has converged to E.

8.2. Numerical test results. The algorithms were tested for a reasonable range of parameters, e.g., for Johnson's algorithm we varied the number of eigensolves between 2^8 and $\approx 2^{13}$. The parameter choices are not particularly important other than forcing each algorithm to take longer to produce a more accurate result; it is the error-cost relationship that we are interested in. The results are shown on a log-log plot in Figure 8.1. A linear fit has been applied so that the asymptotic complexity of each algorithm can be ascertained. This was not done for Uhlig's full algorithm.

Johnson and Johnson using Lanczos with continuation reduce error as predicted in [26], as $\mathcal{O}(m^{-2})$. Uhlig-lite reduces error approximately as $\mathcal{O}(m^{-3})$. The new algorithm reduces error approximately as $\mathcal{O}(m^{-5})$. This is a significant improvement on existing algorithms in terms of asymptotic performance.

For the test matrix used, the path-following algorithm is faster in absolute terms than all existing algorithms for accuracy better than around 10^{-4} (this excludes Uhlig-lite because it does not count the computation time for the "exact" interpolation used). Notably, the performance of Uhlig's algorithm deteriorates and becomes slower than Johnson's algorithm around 10^{-9} accuracy.

Although the asymptotic results should hold in general, some caution has to be advised when attempting to draw conclusions about practical use cases. Factors such as matrix size, matrix structure, regularity of $\partial \mathrm{W}(A)$, and CPU all have an influence on the results. In particular, if a test matrix is structured such that computation of linear solves are appreciably faster than eigenproblem solves, then the performance of the path-following algorithm improves significantly.

8.2.1. Additional details on test implementation. There are some pertinent details concerning the implementation of the numerical tests which deserve further explanation.

As mentioned in subsection 3.3, Uhlig described four methods [46, sect. 2.1] for selecting new edge-points. After tests using Uhlig's `wberell2.m` code [45], method

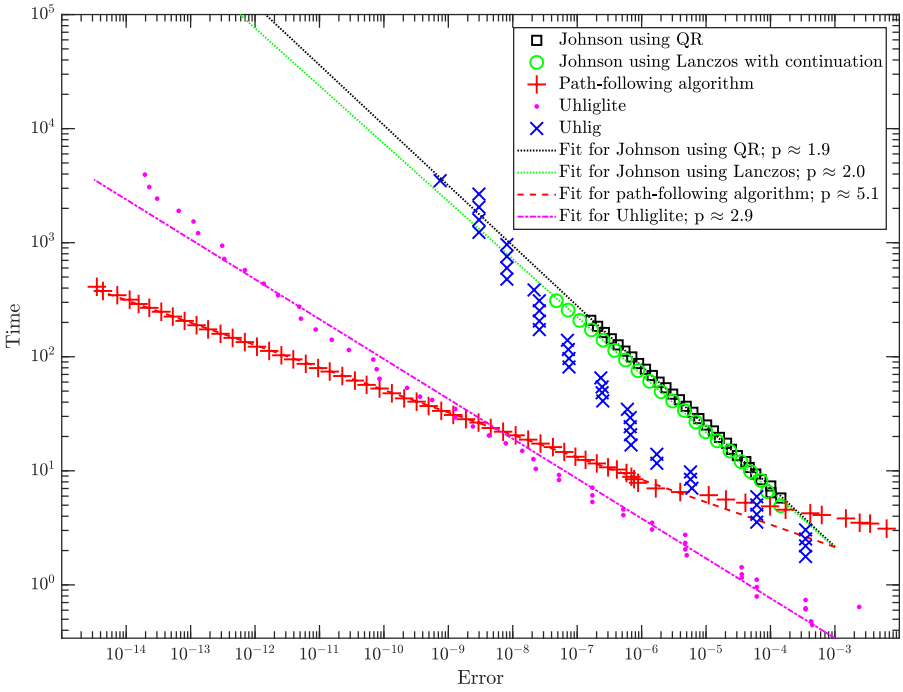


FIG. 8.1. *Log-log plot of computation time against error of the different algorithms. The asymptotic error in Johnson's algorithm is of order $\mathcal{O}(m^{-2})$, Uhlig-lite $\mathcal{O}(m^{-3})$, and our new algorithm $\mathcal{O}(m^{-5})$. Note that Uhlig-lite is a theoretical lower bound for Uhlig where we do not count the interpolation costs and is not in itself a practical algorithm.*

4 was generally the most efficient, and this is what was used in our numerical tests. The number of points per ellipse is selected for adaptively in the `wberell2.m` code and constrained to only those points likely to be on the boundary.

MATLAB's `eig()` (QR) was chosen as the default over `eigs()` (Lanczos) due to it being more efficient for the matrix size used. As seen in the results, this situation changes when continuation can be used but the distinction is marginal.

A matrix size of $n = 250$ was chosen as it was the largest size which could be practically tested with available computing resources: the runtime of each algorithm is relatively quick, but to precisely determine the error required several days runtime on a multicore system. Due to the limitations of `double` word precision used by MATLAB, we could not test any algorithm more accurately than approximately 10^{-14}.

8.3. Eigenvalue crossings. We tested $10,000$ random matrices of size $n = 100$ and the same number of size $n = 10$ and could not record a single eigenvalue crossing. This is in accordance with von Neumann and Wigner's result as described in subsection 7.1. It is, however, trivial to deliberately craft matrices which do exhibit eigenvalue crossings, e.g., the matrix specified by (7.1).

8.4. Discussion. The numerical tests support the claim that the new algorithm has better asymptotic performance than the Johnson, Braconnier–Higham, and Uhlig algorithms. For the matrix tested, the new algorithm was faster in absolute terms for error tolerance better than $\approx 10^{-4}$. For randomly generated matrices, we can be confident that the path-following algorithm will only require one eigenvalue solve

irrespective of the required accuracy. Given that this is the dominant factor in computational cost, the path-following algorithm should compare favorably to other algorithms which are heavily dependent on eigenvalue solves.

9. Conclusions and future work. We have presented a new algorithm for calculating the field of values boundary for complex matrices. At high accuracy, our algorithm is at least an order of magnitude faster than all previous algorithms. There are open questions on finding optimal parameter sets and opportunities for further analysis of nonnormal type 3 matrices which may form the subject of future efforts.

As mentioned in the introduction, we believe that the path-following algorithm we have presented has more generic applicability. Indeed, one of the authors is currently developing a minor alteration of the algorithm for use in calculating dispersive properties of ocean waves. As a result of this ongoing work, it appears our algorithm—with suitable adjustments—has broad applicability for efficient solution of Sturm–Liouville problems parametrized by a single real variable.

REFERENCES

[1] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers* I, Springer-Verlag, New York, 1999, https://doi.org/10.1007/978-1-4757-3069-2.

[2] W.-J. Beyn, C. Effenberger, and D. Kressner, *Continuation of eigenvalues and invariant pairs for parameterized nonlinear eigenvalue problems*, Numer. Math., 119 (2011), pp. 489–516, https://doi.org/10.1007/s00211-011-0392-1.

[3] W.-J. Beyn and V. Thümmler, *Continuation of invariant subspaces for parameterized quadratic eigenvalue problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1361–1381, https://doi.org/10.1137/080723107.

[4] T. Braconnier and N. J. Higham, *Computing the field of values and pseudospectra using the Lanczos method with continuation*, BIT, 36 (1996), pp. 422–440, https://doi.org/10.1007/BF01731925.

[5] R. P. Brent, *Algorithms for Minimization without Derivatives*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[6] W. Cheney and D. Kincaid, *Numerical Mathematics and Computing*, Brooks/Cole, Cengage Learning, Monterey, CA, 2012.

[7] M.-T. Chien and H. Nakazato, *Flat portions on the boundary of the numerical ranges of certain Toeplitz matrices*, Linear Multilinear Algebra, 56 (2008), pp. 143–162, https://doi.org/10.1080/03081080701217745.

[8] M.-T. Chien, L. Yeh, Y.-T. Yeh, and F.-Z. Lin, *On geometric properties of the numerical range*, Linear Algebra Appl., 274 (1998), pp. 389–410, https://doi.org/10.1016/S0024-3795(97)00373-X.

[9] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin, Heidelberg, 1993, https://doi.org/10.1007/978-3-662-02945-9.

[10] M. Crouzeix, *Numerical range and functional calculus in Hilbert space*, J. Funct. Anal., 244 (2007), pp. 668–690, https://doi.org/10.1016/j.jfa.2006.10.013.

[11] C. Davis, *The Toeplitz–Hausdorff theorem explained*, Canad. Math. Bull., 14 (1971), pp. 245–246, https://doi.org/10.4153/CMB-1971-042-7.

[12] W. F. Donoghue, *On the numerical range of a bounded operator*, Michigan Math. J., 4 (1957), pp. 261–263, https://doi.org/10.1307/mmj/1028997958.

[13] J. R. Dormand and P. J. Prince, *A family of embedded Runge–Kutta formulae*, J. Comput. Appl. Math., 6 (1980), pp. 19–26, https://doi.org/10.1016/0771-050X(80)90013-3.

[14] P. F. Zachlin and M. E. Hochstenbach, *On the numerical range of a matrix*, Linear Multilinear Algebra, 56 (2008), pp. 185–225, https://doi.org/10.1080/03081080701553768, English translation with comments and corrections on a paper by Rudolph Kippenhahn.

[15] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice Hall, Englewood Cliffs, NJ, 1977.

[16] I. M. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, Resultants, and Multidimensional Determinants*, 1st ed., Birkhäuser Basel, Basel, 1994, https://doi.org/10.1007/978-0-8176-4771-1.

[17] N. Guglielmi and C. Lubich, *Differential equations for roaming pseudospectra: Paths to extremal points and boundary tracking*, SIAM J. Numer. Anal., 49 (2011), pp. 1194–1209, https://doi.org/10.1137/100817851.

[18] N. Guglielmi and C. Lubich, *Erratum/Addendum: Differential equations for roaming pseudospectra: Paths to extremal points and boundary tracking*, SIAM J. Numer. Anal., 50 (2012), pp. 977–981, https://doi.org/10.1137/120861357.

[19] N. Guglielmi and M. L. Overton, *Fast algorithms for the approximation of the pseudospectral abscissa and pseudospectral radius of a matrix*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1166–1192, https://doi.org/10.1137/100817048.

[20] K. Gustafson, *The Toeplitz-Hausdorff theorem for linear operators*, Proc. Amer. Math. Soc., 25 (1970), pp. 203–204, https://doi.org/10.1090/S0002-9939-1970-0262849-9.

[21] K. E. Gustafson and D. K. Rao, *Numerical Range: The Field of Values of Linear Operators and Matrices*, Springer, New York, 1997, https://doi.org/10.1007/978-1-4613-8498-4_1.

[22] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I, Springer-Verlag, Berlin, Heidelberg, 1993, https://doi.org/10.1007/978-3-540-78862-1.

[23] F. Hausdorff, *Der Wertvorrat einer Bilinearform*, Math. Z., 3 (1918), pp. 314–316, https://doi.org/10.1007/BF01292610.

[24] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991, https://doi.org/10.1017/CBO9780511840371.

[25] C. R. Johnson, *Normality and the numerical range*, Linear Algebra Appl., 15 (1976), pp. 89–94, https://doi.org/10.1016/0024-3795(76)90080-X.

[26] C. R. Johnson, *Numerical determination of the field of values of a general complex matrix*, SIAM J. Numer. Anal., 15 (1978), pp. 595–602, https://doi.org/10.1137/0715039.

[27] T. Kato, *A Short Introduction to Perturbation Theory for Linear Operators*, Springer-Verlag, New York, 1982, https://doi.org/10.1007/978-1-4612-5700-4.

[28] R. Kippenhahn, *Über den Wertevorrat einer Matrix*, Math. Nachr., 6 (1951), pp. 193–228, https://doi.org/10.1002/mana.19510060306.

[29] D. Kressner and B. Vandereycken, *Subspace methods for computing the pseudospectral abscissa and the stability radius*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 292–313, https://doi.org/10.1137/120869432.

[30] P. D. Lax, *Linear Algebra and Its Applications*, Pure Appl. Math., 2nd ed., John Wiley & Sons Ltd, 2007, http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471751561.html.

[31] C.-K. Li, *A simple proof of the elliptical range theorem*, Proc. Amer. Math. Soc., 124 (1996), pp. 1985–1986, https://doi.org/10.1090/S0002-9939-96-03307-2.

[32] S. H. Lui, *Computation of pseudospectra by continuation*, SIAM J. Sci. Comput., 18 (1997), pp. 565–573, https://doi.org/10.1137/S1064827594276035.

[33] S. H. Lui, H. B. Keller, and T. W. C. Kwok, *Homotopy method for the large, sparse, real nonsymmetric eigenvalue problem*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 312–333, https://doi.org/10.1137/S0895479894273900.

[34] M. Marcus and C. Pesce, *Computer generated numerical ranges and some resulting theorems*, Linear Multilinear Algebra, 20 (1987), pp. 121–157, https://doi.org/10.1080/03081088708817748.

[35] B. Moyls and M. Marcus, *Field convexity of a square matrix*, Proc. Amer. Math. Soc., 6 (1955), pp. 981–983, https://doi.org/10.2307/2033121.

[36] F. D. Murnaghan, *On the field of values of a square matrix*, Proc. Natl. Acad. Sci. U.S.A., 18 (1932), pp. 246–248, http://www.pnas.org/content/18/3/246.full.pdf.

[37] P. J. Psarrakos and M. J. Tsatsomeros, *Numerical range: (in) a matrix nutshell*, Mathematical Notes from Washington State University, v. 45 (2002) / v. 46 (2003), 2003, http://www.math.ntua.gr/~ppsarr/nutshell.ps.

[38] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, 2nd ed., Springer-Verlag, Berlin, Heidelberg, 2007, https://doi.org/10.1007/b98885.

[39] R. Raghavendran, *Toeplitz–Hausdorff theorem on numerical ranges*, Proc. Amer. Math. Soc., 20 (1969), pp. 284–285, https://doi.org/10.1090/S0002-9939-1969-0233186-5.

[40] F. Rellich, *Perturbation Theory of Eigenvalue Problems*, Gordon and Breach Science Publishers, New York, London, Paris, 1969.

[41] L. F. Shampine, *Some practical Runge–Kutta formulas*, Math. Comp., 46 (1986), pp. 135–150, https://doi.org/10.2307/2008219.

[42] P. Sirković and D. Kressner, *Subspace acceleration for large-scale parameter-dependent Hermitian eigenproblems*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 695–718, https://doi.org/10.1137/15M1017181.

[43] T. Toeplitz, *Das algebraische Analogon zu einem Satze von Fejér*, Math. Z., 2 (1918), pp. 187–197, https://doi.org/10.1007/BF01212904.

[44] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, 2005, http://press.princeton.edu/titles/8113.html.

[45] F. Uhlig, *Source code from Frank Uhlig's webpage*, http://www.auburn.edu/~uhligfd/m_files/wberell2.m (retrieved Jan. 2017).

[46] F. Uhlig, *Faster and more accurate computation of the field of values boundary for n by n matrices*, Linear Multilinear Algebra, 62 (2014), pp. 554–567, https://doi.org/10.1080/03081087.2013.779269.

[47] J. von Neumann and E. Wigner, *On the Behavior of the Eigenvalues of Adiabatic Processes*, World Scientific, pp. 25–31, https://doi.org/10.1142/9789812795762_0002.

[48] J. von Neumann and E. Wigner, *Über das Verhalten von Eigenwerten bei adiabatischen Prozessen [On the behavior of the eigenvalues of adiabatic processes]*, in German, Physik. Z., 30 (1929), pp. 467–470.